☆ Delete Nodes Greater Than X

Complete the *removeNodes* function provided in your editor. It has *2* parameters:

1. *list*: A reference to a LinkedListNode that is the head of a linked list.
2. *x*: An integer value.

Your function should remove all nodes from the list having data values greater than *x*, and then return the head of the modified linked list.

Input Format

The locked stub code in your editor processes the following inputs and passes the necessary arguments to the *removeNodes* function:
The first line contains *N*, the number of nodes in the linked list. Each line *i* (where $0 \le i < N$) of the *N* subsequent lines contains an integer representing the value of a node in the linked list. The last line contains an integer, *x*.

Constraints

- $1 \le N, x \le 10^5$
- $1 \le list_i \le 10^5$, where $0 \le i < N$

Output Format

Return the linked list after removing the nodes containing *values* > *x*.

Sample Input 0

```
5
1
2
3
4
5
3
```

Sample Output 0

```
1
2
3
```

Sample Input 1

```
5
5
2
1
6
7
5
```

Sample Output 1

```
5
2
1
```

Explanation

Sample Case 0: $N = 5$, $x = 3$

$list = 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$

After removing the nodes having *value* > 3, *$list = 1 \rightarrow 2 \rightarrow 3$*.

Sample Case 1: $N = 5$, $x = 5$

$list = 5 \rightarrow 2 \rightarrow 1 \rightarrow 6 \rightarrow 7$.

After removing the nodes having *value* > 5, *$list = 5 \rightarrow 2 \rightarrow 1$*.

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour.

Start tour

Original code Java 7

```java
import ↔;

public class Solution {
    public static class LinkedListNode{
        int val;
        LinkedListNode next;

        LinkedListNode(int node_value) {
            val = node_value;
            next = null;
        }
    };

    public static LinkedListNode
_insert_node_into_singlylinkedlist(LinkedListNode head,
LinkedListNode tail, int val){
        if(head == null) {
            head = new LinkedListNode(val);
            tail = head;
        }
        else {
            tail.next = new LinkedListNode(val);
            tail = tail.next;
        }
        return tail;
    }

```

VMWare Challenge UR Propel

45m : 37s
to test end

1

2

3

4

```
/*
 * Complete the function below.
 */
/*
For your reference:
LinkedListNode {
    int val;
    LinkedListNode *next;
};
*/

    static LinkedListNode removeNodes(LinkedListNode list, int x) {
        LinkedListNode dummy = new LinkedListNode(0);
        dummy.next = list;
        LinkedListNode curr = dummy;

        while (curr.next != null) {
            if (curr.next.val > x) {
                curr.next = curr.next.next;
            } else {
                curr = curr.next;
            }
        }

        return dummy.next;

    }


    public static void main(String[] args) throws IOException{↔}
}
```

Line: 54 Col: 27

Test against custom input

Run Code

Submit code & Continue

(You can submit any number of times)

VMWare Challenge UR Propel

45m : 37s
to test end

Compiled successfully. All available test cases passed!

1

2

3

4

Test Case #1:	✓	Test Case #9:	✓
Test Case #2:	✓	Test Case #10:	✓
Test Case #3:	✓	Test Case #11:	✓
Test Case #4:	✓	Test Case #12:	✓
Test Case #5:	✓	Test Case #13:	✓
Test Case #6:	✓	Test Case #14:	✓
Test Case #7:	✓	Test Case #15:	✓
Test Case #8:	✓	Test Case #16:	✓

Testcase 1: *Success*

Your Output

```
1
2
3
```

Expected Output

```
1
2
3
```

Testcase 2: *Success*

Your Output

```
5
2
1
```

Expected Output

```
5
2
1
```

Testcase 3: *Success*

Your Output

```
Output hidden
```

Testcase 4: *Success*

Your Output

```
Output hidden
```

Testcase 5: *Success*

Your Output

```
Output hidden
```

Testcase 6: *Success*

Your Output

```
Output hidden
```

Testcase 7: *Success*

Your Output

```
Output hidden
```

Testcase 8: *Success*

Your Output

```
Output hidden
```

Testcase 9: *Success*

Your Output

```
Output hidden
```

Testcase 10: *Success*

Your Output

```
Output hidden
```

Testcase 11: *Success*

Your Output

```
Output hidden
```

Testcase 12: *Success*

Your Output

```
Output hidden
```

Testcase 13: *Success*

Your Output

```
Output hidden
```

Testcase 14: *Success*

Your Output

```
Output hidden
```

Testcase 15: *Success*

Your Output

```
Output hidden
```

Testcase 16: *Success*

Your Output

```
Output hidden
```